COMPUTER STRATEGIES
FOR THE GAME OF QUBIC

by

WILLIAM GEORGE DALY JR.

S.B., Massachusetts Institute of Technology
(1958)

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February, 1961

Signature of Author _____
Department of Electrical Engineering, January, 1961

Certified by _____
Thesis Supervisor

Accepted by _____
Chairman, Departmental Committee on Graduate Students

COMPUTER STRATEGIES

FOR THE GAME OF QUBIC

by

WILLIAM GEORGE DALY JR.

Submitted to the Department of Electrical Engineering on January 10, 1961 in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering.

ABSTRACT

This thesis describes a program which was written for the TX-0 computer to play the game of Qubic. The game is a three dimensional form of Tic-Tac-Toe with four squares on each side. The game is finite but at present unsolvable, as about 10^{25} different games are shown to exist.

The program uses rather standard lookahead and evaluation techniques, but employs a forcing lookahead to a depth of 12 to detect winning positions. This strategy is a major factor in the program's success.

A system of "opponent analysis" is also used in which the program is able to distinguish between different human players. The program will drastically modify its line of play in each game by consulting a record of the results of previous games against this opponent. Over a hundred games were recorded with the machine winning over 80%.

Appendices include: a discussion of symmetry and a derivation of all symmetries of the game; a calculation of the total number of possible positions and possible games; a mathematical description of the game as employed in the program; a discussion of forcing win situations and a description of all "three in a plane" possibilities; and a move by move analysis of several actual games.

Thesis Supervisor: Dean Norman Arden
Title: Associate Professor of Electrical Engineering

ACKNOWLEDGEMENT

I wish to thank Professor Arden, who encouraged and guided this work, and my wife, who spent many long nights in the computer room providing opposition for the machine.

TABLE OF CONTENTS

I. History of the Problem

With the development of high speed electronic computers in the last ten years, a great deal of work has been done programming these computers to play various types of games. A major reason for the interest in this work is the fact that the computation involved is symbolic rather than numeric. Games provide a good vehicle for the study of symbolic computation since they have a system of rules for the computation to follow and a well defined goal which allows an evaluation of the effectiveness of the process and a comparison of the corresponding effectiveness of human players.

Much of this work has centered around the class of finite, no chance, perfect information games. The characteristics of these games are:

a) There is a definite rule for defining the "end" of the game. This end must be reached after a finite number of moves and the "result" of the game must then be obtainable.

b) There are no decisions in the game which are determined by chance, no rolls of dice, etc.

c) Neither player can conceal anything about decisions he has already made. Each player has perfect information about the state of the game to date.

Many programs have been written to play such trivial games as Nim and Tic-Tac-Toe, but most of the important work has been on the games of chess[1-6] and checkers.[7-8] The general pattern for most of this work was first advanced by Wiener and Shannon in the late forties. The most successful operating program to date is probably the one by Samuel which plays checkers.[8]

II. General Strategies

Nearly all of the strategies which have been developed for this type of game center around two basic techniques, lookahead and position evaluation. The technique of lookahead is essentially the generation of some portion of the total move tree which starts at the present position. In its pure form the lookahead would run to the end of the game, and the best move for the present position could then be determined by a process of "minimax", where it is assumed that the opponent will choose among his altermative moves so as to "minimize" the chance of the player's winning, and the player will choose among his moves so as to "maximize" his chance of winning.

In a simple game like Tic-Tac-Toe it is easy to see how this process could be carried to the extreme, and programs have been written to play Tic-Tac-Toe in exactly this manner. As the game becomes more complicated, however, the amount of computation involved increases rapidly, and soon becomes completely impractical for present day computing devices. Samuel has estimated that in checkers, for instance, a machine investigating moves at the rate of three hundred million per second would still require 10^{21} centuries to complete the lookahead from the first move.[8]

To solve this problem the technique of position evaluation has been used to reduce the size of the move tree investigated. Position evaluation is usually some type of relationship which allows a given position to be assigned a numerical value which is approximately proportional to the probability of eventually winning the game, given

imperfect but nearly equal play by the two opponents during the remainder of the game.

Position evaluation formulas take one of two forms, an equation type of relationship or a table listing. To contrast the two methods consider the games of Nim and Tic-Tac-Toe. In Nim a simple formula exists for counting the number of pieces in each row and then comparing these sums to determine if the position is "winning" or "losing". The program need only try all possible immediate moves and apply the formula to the position generated. As soon as a "losing" position is detected the program can make the move which generated it, since the resulting position will be "losing" for the opponent. If none of the possible moves result in a losing position, however, then the present position must be losing and it will be impossible for the program to make a move which will insure a win. In this case the program should pick the move which will make it "hardest" for the opponent to win, but this may be hard to define and will be discussed later.

In Tic-Tac-Toe the total number of possible board positions is small enough that the entire move tree could first be calculated and then worked through from the end, as suggested by Bellman,[9] so that each position could be classified as winning, losing, or drawn. The program could then proceed as above, but would now look up the resulting board position in the table rather than use a formula to determine its value. This process could be carried out for the entire game before play started and the best possible move stored with each position rather than its value. Now the program need only look up the present position and make

the indicated move.

In a more complicated game the perfect formula is not available nor is it practical to list all possible board positions which may be encountered. In this situation it is necessary to approximate the perfect formula in some way, usually a linear sum of some abstract properties of the game which are believed to define a "good" position. Samuel has produced some interesting results by letting his program choose the quantities to be used in the sum and modify the coefficients by which they are multiplied as a function of the success of the program. In another program, he was able to combine the two forms by saving the evaluations which had been made of encountered positions through the lookahead and minimax process.[8]

The most important factor in making a system of remembering specific board positions work is the way in which they are stored. If the entire memory of positions must be searched each time the table is consulted, the amount of time per move will become excessive, long before a sufficient number of positions are collected. One way to set up an efficient storage system is to define several properties of positions and use these values as "addresses". A typical set of these properties for Qubic might be the number of pieces on the board, number of pieces in the 16 core squares, and number of two in a row, Values for these parameters could be calculated quite easily and could even be updated along with the state of the board as moves were made. If the range of each of these values was only ten, the search time would be reduced by a factor of about 10^{-3}. The actual system of such addressing must depend mainly

on the machine, the game, and the ingenuity of the programmer. In order to use these techniques, however, it is necessary to have some sort of high volume, medium speed memory, such as magnetic tape, drum, or disk.

Another valuable form of position evaluation is the "move generator". Although in a specific program the move generator routine may actually generate moves from the present board position, it can be considered to be a simple lookahead of all possible moves and then the application of a rough evaluation formula to the resulting positions. This evaluation will define some subset of the set of all possible moves which are "better" than the rest and are worth examining in more detail by the general lookahead routine. This allows a great deal of computational time which would be wasted on the exploration of bad moves to be used in looking farther ahead on promising or dangerous situations.

The decision as to exactly how many moves, or "plys", the lookahead should go before the position evaluation formula is used, is a very critical parameter in the performance of the program. Shannon has pointed out the danger of applying an evaluation formula to a position which is not quiescent, where a trade is in progress or where forcing moves exist.[1] The best technique seems to be a dynamic decision in which another type of evaluation is made to determine if the direction of play is not promising and should be discontinued or if the present position is one of transient nature and the lookahead must continue farther in order to determine the actual value. This involves following forcing situations to great depth than normal play, but this is not too costly since the replies of the opponent

are quite limited and the size of the tree does not grow rapidly.

This suggests another interesting technique, that of forcing lookahead. A move generation routine is used which generates only forcing or near forcing moves, and these moves are followed to much greater depth than would be possible in a normal lookahead, since the number of moves to be investigated will be smaller and the replies of the opponent quite limited. In some situations it may be possible to allow the depth to be the end of the game, or some very large maximum such as 15 or 20, when normal depth limits would run from three to six.

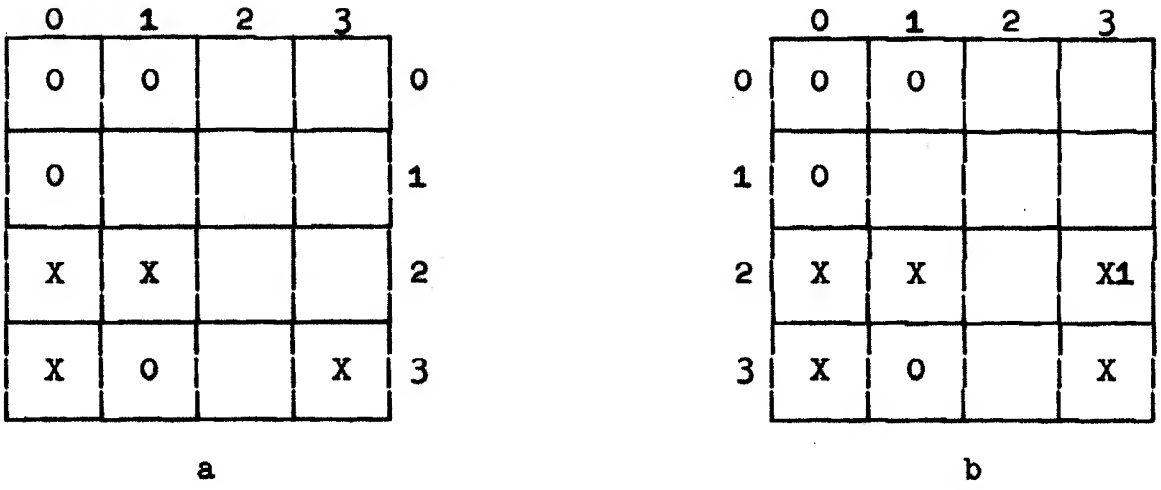
III. The Strategies Employed

From the numbers calculated in Appendix Two, 10^{26} different board positions and 10^{24} different games, it is clearly impossible to solve the game by calculating the entire move tree from any point or by calculating the entire move tree once and saving an evaluation of every position. In order to develop some good strategies, it is interesting to follow the philosophical concept of Dynamic Programming and consider the possible ways the game can end.

An "intelligent" end of the game must come when one of the players can pick a square in such a manner as to generate a three to nothing situation on two different lines, of which only one can be blocked. To work back one step farther, this situation can only be reached intelligently if two different pairs of intersecting lines can be developed with one move, or if a sequence of forcing moves, generating three in a row which must be blocked, can be executed which will result in one such pair of lines and the move.

A simple forcing situation of this type is shown for a single plane in Figure I. In the initial board position, Figure Ia, there is no move for X which would develop two three in a rows and an immediate win. Figure Ib shows the first move by X as 23, which <u>forces</u> O to block at 22, as in Figure Ic. This allows X to move at square 03 and a win is assured, since O must block at both 12 and 13. In this situation the order of X's moves is not important, if he had moved first at 03 and then at 23 the same situation

Figure I: Simple Forcing Configuration



would have developed. This is not always the case, however, for if O had occupied 11 as well, the move at 03 would have allowed O to force at 13 and X would have been unalbe to win, while if X had taken 23 first, the win would still have been possible.

In most games the play seems to follow the forcing type conclusion, and most strategical principles which human players employ deal with forcing situations, especially in a particular plane. (Although these principles were not used directly they are discussed in Appendix Four.) For this reason it was decided to employ a forcing lookahead routine which would run to great depth over all possible three in a row situations.

A depth of ten was used as an upper bound and seemed to be sufficient, as most wins were reached at around the six or seven level. Although the upper limit for the amount of time this calculation could take is around six months, and an actual situation was once encountered which would have taken eight hours, the usual time is between ten and twenty seconds. Since the routine was so fast it was made to run with increasing upper limits, first one, then two, etc., until the final limit of ten was reached. Because of this it could be assured that the resulting sequence consisted of the minimum number of moves and that all were essential, extremely important facts when the routine is used for defensive purposes.

The problem of what to do if no win is found has a rather difficult aspect, for if we accept the principle of only evaluating quiescent positions, it would be necessary to use the forcing lookahead on every position which is

evaluated to first determine if a win exists. It therefore seemed most practical to go in the direction of a short lookahead and as good an evaluation formula as possible.

When an evaluation formula is used to decide which possible move is the "best", it is really only necessary to know the change in the state of the game produced by each, rather than the total state produced. All moves start from an equal position, and whichever one produces the biggest positive change in position will therefore be the one which results in the highest total evaluation.

When a square on a particular line is chosen, that line can have an effect on the change of the state of the game only if one of the following five conditions exists:

a) A two in a row is converted to a three in a row and the opponent will be forced to block.

b) A one in a row is converted to a two in a row and a possible forcing situation is established.

c) A previously unoccupied line is developed into a one in a row.

d) An opponent's one in a row is blocked.

e) An opponent's two in a row is blocked and one of his possible forcing situations is removed.

The evaluation formula used was simply a linear sum of the number of each type of "live" line, and the "change" function was therefore a linear sum of the number of each of the above five changes produced.

In order to obtain a good set of coefficients by which to multiply each of the parameters, a number of "machine versus machine" and "machine versus good player" games were conducted with different sets. The ratio of the value for

generating a two in a row to the value for generating a one in a row was found to be about $(40{:}10)_8$. When blocking a two in a row or a one in a row the ratio dropped to $(40{:}6)_8$. The value of forcing when there was no possible win was quite low, about $1/40_8$ the value of generating a two in a row. These values seemed fairly accurate and were about the best against all types of opponents. The ratio of offense to defense, or of generating a two in a row to blocking a two in a row, had a great effect on the quality of play, and when describing the final coefficients it is convienent to include this ratio, r, as $(r{:}40r{:}10r{:}6{:}40)_8$, for the order given above.

Against good players it was necessary to make the value of r about one-fourth, which made the machine play extremely conservatively and concentrate more on defense than offense. The main reason for this was probably that almost all games were played with the machine going second, which definitely put it on the defensive. In addition the player who has just moved is on the defnesive to the extent that he has lost the initiative.

The general strategy employed is to first run the board through a procedure which checks to see if the game has been won or lost, if it can be won, or if a block must be made. The flow of this process is shown in Figure II and of the complete process in Figure III. If none of these trivial situations exists, the machine performs the forcing lookahead to see if the game can be won. If the game cannot be won the board is reversed and the lookahead is again performed, this time for the opponent. If a win is detected for the opponent the lookahead acts as a move

Figure II: Initialize Flow

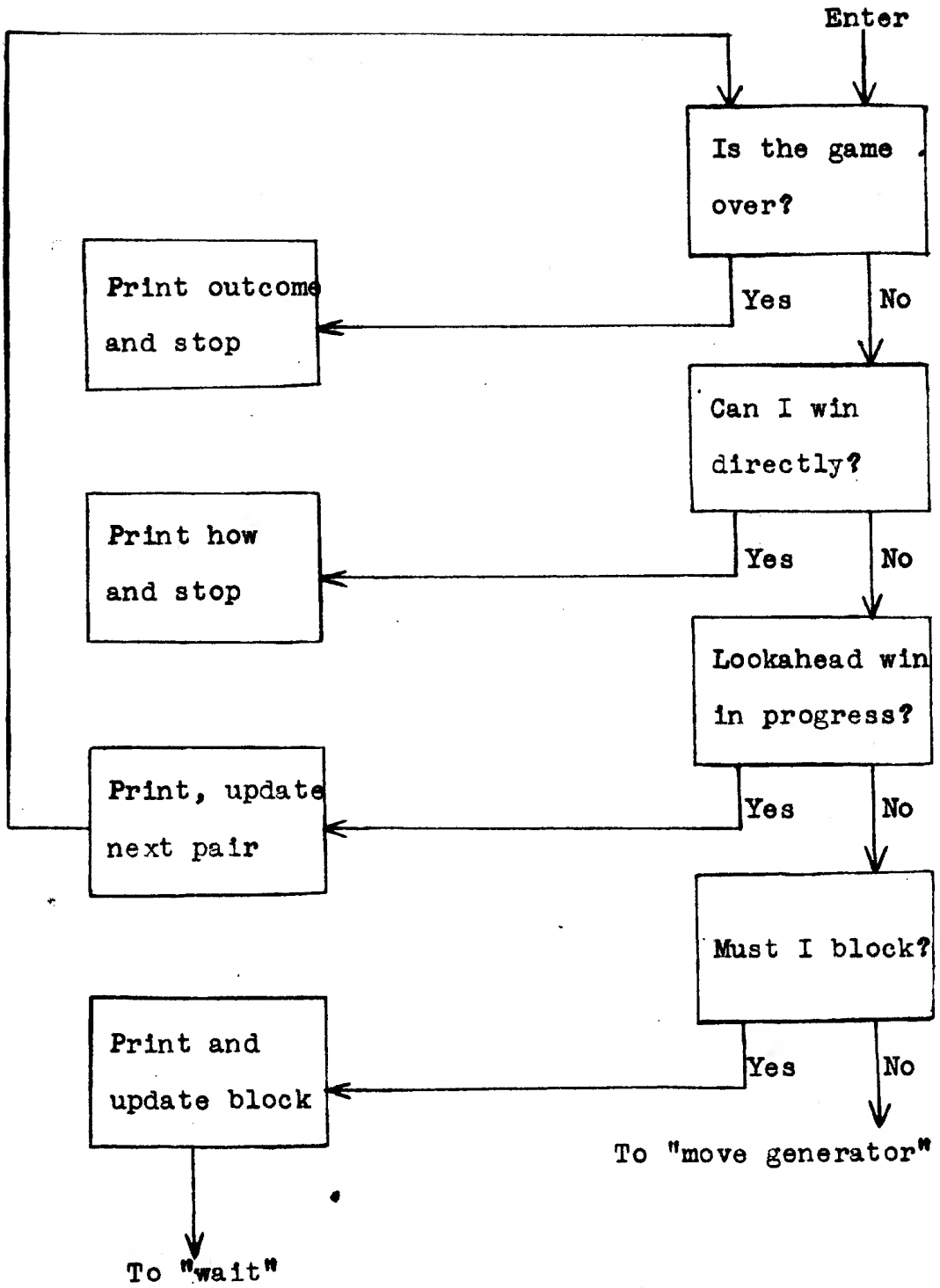
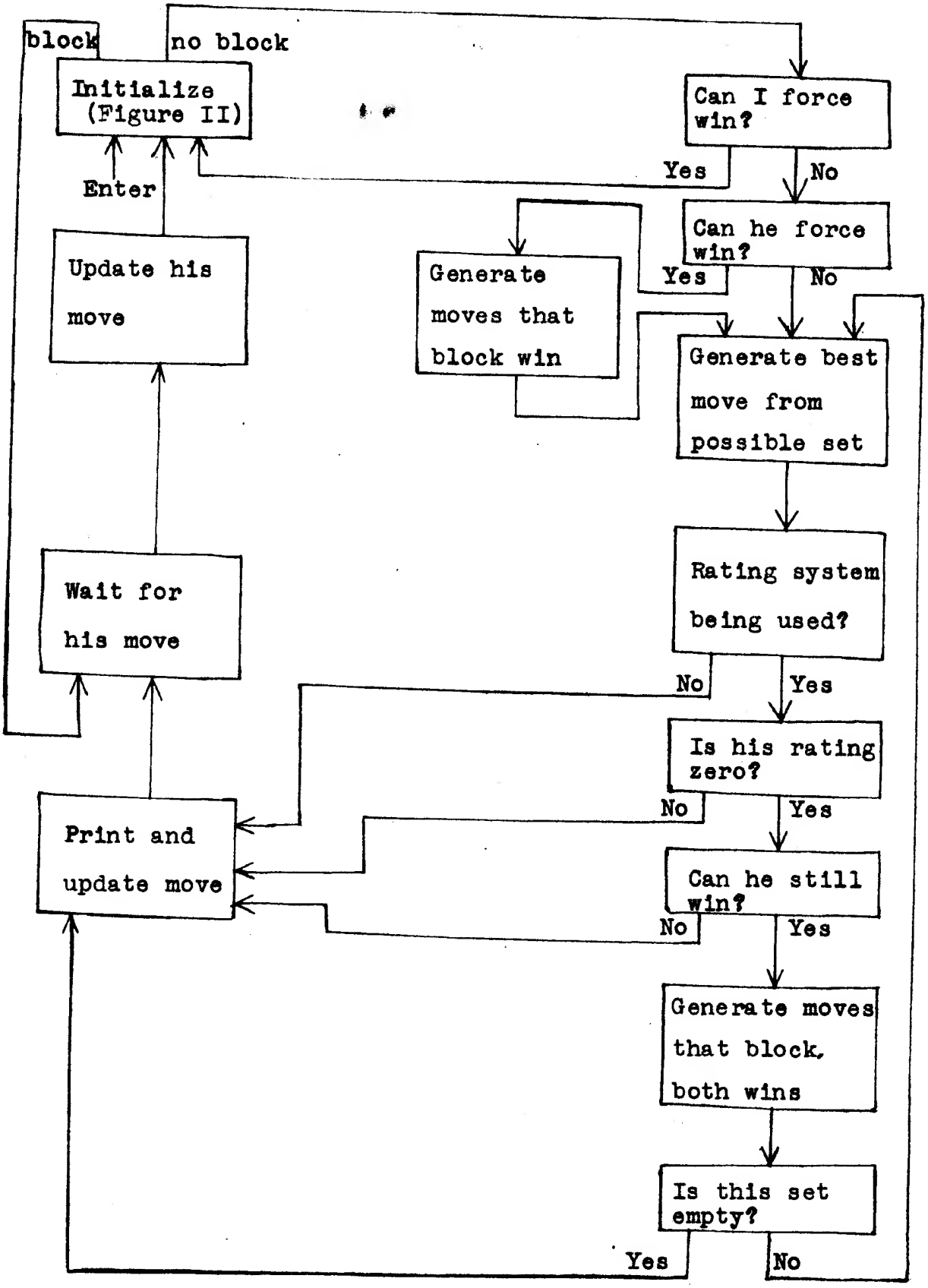


Figure III: Strategy Flow



generator and the final move must come from the set of moves which will block that win. A simple lookahead of one move is then performed over all possible moves and the previously discussed evaluation formula is applied. The move with the highest evaluation is then chosen and updated. The lookahead is again applied and if the opponent can still win the move is discarded and the set of possible moves is reduced to those which will block both wins. Another move is then generated from this set and the above process repeated. This continues until a move is found which will keep the opponent from winning or it is found that there are no moves which will keep the opponent from winning. The chosen move is then printed and the machine stops and waits for the opponent's move.

IV. Opponent Evaluation

During the discussion of general strategies it was noted that when the machine finds itself in a losing position it should choose its next move in such a manner as to make it "hardest" for the opponent to win. This problem has previously been handled by choosing the move which made the most improvement in the state of the game, as defined by the evaluation formula. An alternative technique will now be considered in which the move is chosen on the basis of weaknesses of the specific opponent which have been determined by making an analysis of previous play. This technique will also be used when a winning situation exists in order to develop the win as rapidly as possible, or in a drawn situation in order to have the best chance of converting this into a win. An attempt is made to introduce a motivation into the program to win as quickly as possible and to spend a minimum amount of computational time to defeat the specific opponent.

It is certainly possible to conceive of quite elaborate schemes which could be used for saving information about a particular player in order to determine the best approach to take when playing against him again. Entire games might be saved in order to obtain an estimation of how he will react to a given situation. Most human players seem to visualize the game in terms of individual planes, and an analysis of past games could be conducted in order to see which planes a particular player can visualize most easily, in order to force play into other planes and place him at a disadvantage. Rather than build up such an elaborate

system it was attempted to keep the evaluations as simple as possible so that the general concept could be studied more than the specific rating system employed.

After considering various games against different quality opponents, it appeared that the major difference between good and bad players can roughly be described as a difference in the length of their forcing lookaheads. Poorer players tend to "overdevelop" their position,,they pass up situations where they could force to a win in order to develop a more powerful situation in which the sequence of moves is shorter and the geometry easier to visualize. Quite often this delay will allow the machine to take advantage of a more complex situation and win before the player has a chance to use the powerful situation which he has developed. This also indicates that the average player tends to undervalue the machine's position and wait until it is too late to block it.

When playing against this type of opponent it is very profitable to increase the ratio of offense to defense, emphasize quick development of the machine's position at the expense of allowing the opponent to have possible winning situations which it is assumed he will not be able to capitalize on. In order to achieve this end, four basic parameters of the program were made functions of the opponent's rating.

The actual system used consists of assigning each player, (up to 64 players can be remembered at once), a player number and an octal rating in the range 00 to 77.* The player is allowed to choose his own initial rating and play begins.

*There is no theoretical upper limit, but no one has ever continued to play when his rating went above 70.

After each game the rating is divided by two if the opponent wins. If the machine wins and takes less than ten moves before it starts the forcing sequence, the rating is increased by ten minus the number of moves required. If more than ten moves were needed to establish the winning position the rating is unchanged.

The rating has the following effects on the strategy used by the machine:

a) If the rating is 30 or less and the opponent plays first and takes a core square, the machine will make the "reflected" core response. This move has proven empirically to be the best opening for the second player. It is generated by partial adding 25_8 to the first player's move. If the rating is greater than 30 or if the machine goes first or if the opponent does not pick a core square, the first move is chosen in the normal manner.

b) If the rating is greater than 00 the defensive lookahead after a move has been chosen is discarded. The move is immediately updated and printed. (Note Figure III.)

c) The upper limit of the length of the offensive forcing lookahead is set at 12_8 minus the rating divided by 10_8, with a lower limit of five. The upper limit of the defensive forcing lookahead is the offensive limit minus three.

d) The value of r in the evaluation formula is set at the rating plus 10_8 divided by 40_8.

V. Results

After an initial period of debugging, the program played several hundred games against opponents of varying degrees of ability. One of the most interesting results was that no tie game was ever played, nor was anyone ever encountered who had seen a legitimate tie game played. It is not difficult to exhibit a board position which must result in a tie, for instance if red occupies squares 00, 06, 13, 15, 17, 23, 25, 30, 35, 36, 41, 47, 52, 53, 54, 62, 63, 64, 71, 72, and 74; and white occupies squares 02, 03, 04, 11, 12, 14, 21, 27, 32, 33, 34, 43, 45, 50, 55, 56, 60, 66, 73, 75, and 77, then the ultimate outcome of the game must be a tie, despite the fact that each player has made only 21 moves, since both players hold at least one square on every line. Whether a position like this could be reached in an actual game with "good" play by both players is not known. There are several lines in the above game where a three to one situation exists and it may have been necessary for one or both of the players to pass up a direct win in order to reach this state. There is a clear indication that if the game is not a draw then the first player should win, but the strategy which would insure this has not been found.

In most of the games played the machine went second, and the following results are a representative sample of games played against all types of opponents:

First Move	Machine Won		Player Won With Rating Of		
By	Games	%	40-77	01-37	00
Machine	19	95%	1	0	0
Player	78	78%	8	14	1

A fairly good player seemed to reach an equilibrium rating of about ten and split about even with the machine at that rating. Only two players besides the author were able to beat the machine when they went first and had a rating of 00. These wins were developed by a trial and error procedure, in which the players tried different combeinations of moves until they found a weakness in the machine's strategy. There is nothing particularly spectacular about these games and all three players needed the records of previous games in order to repeat the win.

A study of these results showed a general pattern where the machine and the player performed about equally well in the early game, until a board position was reached from which both could force to win and the player had the move. These wins were generally quite complicated, especially for low ratings, and the player usually missed them while the machine did not.

In order to make a better evaluation of the machine's opening strategy and the value of the lookahead, a routine was written which allowed the player to use the lookahead facility of the machine for himself. Immediately after making its move the machine interrogates a switch on the console, bit one of the TAC, and if the information is requested it will check the state of the game to see if any of the following situations exist:

a) The machine has three in a row which the player must block.
b) The player can force to a win from the present position.
c) The machine can force to a win from the present position.

If any of these conditions exist the machine prints out the fact and waits for the player's next move.

The block and loss facilities did not seem to do a great deal of good, but the win information had a drastic effect on the results. The percentage of wins by the machine when the player went first dropped to about 20%, although most of these losses came for ratings in the range of 20-50. For ratings below 20 the machine was still tough to beat, but above 40 it almost never won, since one of the basic reasons for motivating this type of play, that the player would be fooled about when he could win, was removed.

It is interesting that only the information that a win does exist is necessary, not what that win is. It is usually fairly easy to see what forces will develop the position, and if these moves can be made safely the complexity and length of the win are reduced to a point where it can be easily seen. A knowledge of the "three in a plane" forcing rules given in Appendix Four is especially valuable in this type of play.

In the few games that were played with the machine going first the loss facility became much more valuable, since the machine usually reached a winning position sooner than the player. Using this facility I was able to defeat the machine when it went first and I had a rating of 00, but it took about 40 trial and error games before this could be accomplished. Again there isn't any general strategical principle that underlies the game, it just exploits a weakness in the machine's strategy, and it took about ten more trial games to reconstruct it without the record of the previous win.

Appendix Six contains machine printouts for several representative games. Those included are:

a) Player moves first with rating of 40 and helper and wins.

b) Player moves first with rating of 40 and no helper and loses.

c) Player moves first with rating of 00 and helper and wins.

d) Machine moves first with player rating of 00 and helper and loses.

e) Machine plays machine for ratings of 00 and first player wins.

VI. Conclusions

Although the program compiled a rather impressive won-lost record, it doesn't seem that it can be classified as an expert. One of the major problems in this respect is that there are very few human "experts" on Qubic and it is hard to judge the play of the machine by human standards, as is possible in chess and checkers.

The forcing lookahead is certainly a very powerful strategy and is probably necessary in some form or another in any advanced Qubic routine. The lookahead program could probably be written to run slightly faster, especially on a machine with indexed addressing, but the time required was hardly ever excessive and the depth limit of ten was sufficient. As long as a forcing lookahead can be made to run this fast the introduction of the forcing rules discussed in Appendix Four would be of little help, since all of these cases are covered in the general forcing situation.

The program was certainly weakest in opening strategy and that would be the place to devote any further effort. The evaluation formula was extremely successful, considering its simplicity. The main weakness of the formula is the fact that it fails to consider the state of the lines which intersect the lines it is blocking or developing. As was noted in Appendix Four, one of the main advantages in making a force is that the forcing piece also develops a new forcing possibility on an intersecting line which previously had only one square occupied. By writing a routine which investigated all lines which ran through the open squares on the

line which was being developed and by modifying the "score" awarded for the line as a function of the state of the lines which intersect it, the evaluation formula could be improved a great deal.

Extending the length of the general lookahead would also help, and it could probably be run to a depth of two or three without taking an excessively long amount of time. The forcing rules might be a big help in this respect as they would constitute a rough forcing lookahead which could be applied quickly at the end of the regular lookahead. It might also be profitable to attempt to direct the development of the machine away from situations which are covered by the three in a plane rules, which are easy for the human player to visualize, and towards more complicated sequences which take full advantage of the vertical diagonals.

Although minimax is certainly a valuable concept, it would seem that when playing against an opponent who is not able to "minimize" perfectly it would be wise to take this into consideration and attempt to exploit his weaknesses. This can be done on two levels, either for the entire class of opponents, as suggested above, or differently for each individual player. To the best of my knowledge this is the first game playing program which has attempted to distinguish between different players and to play differently because of this, and it certainly seems to be an area with great possibilities. When more sophisticated techniques are developed for using computers to solve decision problems in which people are involved, it would be foolish not to give the computer as much information as possible about the weaknesses and traits of the participants so that they can be exploited

as much as possible.

Probably the best area for writing a program which learns about Qubic is the development of the table type evaluation formula which saves encountered positions. If a longer lookahead were used and a better evaluation formula developed for the regular move generator, the evaluations made in that manner could be saved. All positions which are evaluated by the forcing lookahead could be saved with the information as to whether or not they were winning. Saving positions is very difficult on a machine which does not have magnetic tape drives, and the TX-0 did not at the time this program was written. For this reason a learning scheme of this type was not undertaken. Qubic does not lend itself well to the "generalized" type of learning suggested by Samuel because of the limited number of abstract strategies and concepts available.